Adaptive Object Detection Based on Modified Hebbian Learning

Yong-Jian Zheng and **Bir Bhanu**
College of Engineering
University of California, Riverside, CA 92521-0425
zheng@constitution.ucr.edu, bhanu@engr.ucr.edu

Abstract

This paper focuses on the issue of developing self-adapting automatic object detection systems for improving their performance. Two general methodologies for performance improvement are first introduced. They are based on parameter optimizing and input adapting. Different modified Hebbian learning rules are developed to build adaptive feature extractors which transform the input data into a desired form for a given algorithm. To show its feasibility, an input adaptor for object detection is designed as an example and tested using multisensor data (optical, SAR, and FLIR). Test results are presented and discussed in the paper.

1. Introduction

This paper is motivated by the increased demand for new theories and methodologies to characterize and improve system performance [3, 7, 8] and to minimize the effort needed for the development of robust systems for practical applications of computer vision and pattern recognition. The original contribution of this paper is the idea that the performance of a given algorithm can be improved by adding an adaptor between the input data and the algorithm.

When a pattern recognition and computer vision system is not performing satisfactorily, generally, we find that the new input data consist of variations which are not modeled and considered during the design process of the system. To show this, four synthetic images with different grades of "complexity" are generated (see the first row in Figure 1) and used as the input data to an algorithm which groups each image pixel into two classes (target or background). The algorithm is developed based on the K-Means principle and K is set to 2 for the test. The second row in Figure 1 shows the performance of the image thresholding algorithm. It can be seen that the algorithm performs well if the input image is ideal or just contains a normally distributed random perturbation (cases **a** and **b**). This is because this perturbation is considered and modeled in the algorithm. Actually each algorithm has its own perturbation model and can, therefore, deal with some input perturbations. However, if the input perturbation is unexpected and does not fulfill its model in the algorithm, the algorithm cannot produce reasonable results (see the results for cases **c** and **d**).

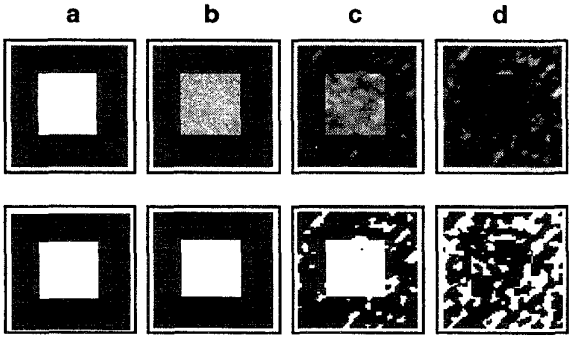

Figure 1. The performance of an image thresholding algorithm.

2. Parameter Optimization or Input Adaptation

Generally speaking, there are two methodologies for adaptation. The first methodology is based on the consideration that some algorithms and systems have certain controllability and their performance can be improved by tuning their parameters [1]. To find the best parameter set for the given input data a learning and optimizing process is usually required. This parameter optimizing oriented methodology, as shown in Figure 2, employs different parameter set for different input data in order to obtain the optimal output. However, this methodology suffers from some inherent shortcomings:

- It is driven by both the input data and the output data. It has to have an off-line learning phase. This means that the correspondence between the input data and the appropriate parameter sets should be first established during an off-line learning process before the algorithm can possess the required adaptability. To perform the off-line learning requires good and enough data samples.

*This work was supported by ARPA, AFOSR grants F49620-93-1-0624, F49620-95-1-0424 and MDA972-93-1-0010. The contents of the information do not necessarily reflect the position or the policy of the U.S. Government.

1015-4651/96 $5.00 © 1996 IEEE
Proceedings of ICPR '96

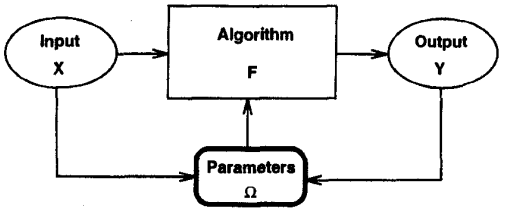

Figure 2. Parameter optimizing methodology for performance improvement.

- In order to use the trained algorithm, information about the possible category of the input data is needed before the appropriate parameter set can be switched on. This means that the trained algorithm works only with an additional input identifier which triggers the corresponding parameter set. Certainly the design of such an identifier is as hard as that of the algorithm itself.
- The performance of an algorithm cannot always be improved by optimizing the parameter set because the gradients of objective functions of some algorithms with respect to their parameters are too small. Not all algorithms can be improved by using this methodology.

Due to these shortcomings, the potential application of this methodology to a real-world problem is limited, although some research has recently been done in this area [1].

The second methodology for performance improvement is based on the observation that most algorithms would perform well if their input data are "friendly", as discussed above. Thus, the performance of almost all commonly used algorithms can be improved by adding an adaptor between the input data and the algorithm (see Figure 3). An ideal adaptor should automatically judge the input data, provide the desired input data to an algorithm, and learn something from this process in order to improve itself in the future.

In comparison with the parameter optimizing based methodology, the input adapting methodology has some positive features such as:

- It is suitable for almost all algorithms because the desired input data (not always the perfect input data) always exists for a given algorithm.
- It is driven only by the input data. So it can work both on-line and off-line. This is very important for real-world and real-time applications.
- It makes it possible to combine some simple, ready-made available algorithms to build vision systems that exhibit high level of performance. Without adding adaptors, these simple algorithms may be unreliable for applications, although they may have simple structures and may not be time consuming.

3. Representations Versus Salient Features

Pattern recognition and computer vision can be thought of as a multistage process of representation transformation from an im-

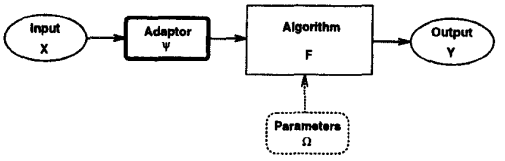

Figure 3. Input adapting methodology for performance improvement.

plicit raster image of a given scene to an explicit map describing the meaning of that scene. Each two adjacent stages are linked by algorithms which transform an input representation to an output representation. As mentioned above, many commonly used algorithms are only designed to deal with those input representations which are relatively "friendly". This means that these input representations contain less spurious and erroneous information and have more explicitness. To distinguish these desired representations from others, we define them as *salient features*. So, salient features are those input representations which possess some "nice" properties, and therefore, desired by a given algorithm. To keep an algorithm's performance high even if the input representations are not so "friendly", adaptors are needed which transform the input representations to some salient features. Thus, an adaptor can also be regarded as a salient feature extractor. The key issue in input adapting methodology for performance improvement is how to design an adaptor or feature extractor for each algorithm at each stage of the representation transformation.

3.1. Optimal Feature Extraction

From a mathematical viewpoint, feature extraction is a transformation from a m-dimensional input representation $\mathbf{x}$ to a n-dimensional output representation $\mathbf{v}$, so that $n \leq m$ and for each $v \in \mathbf{v}$ the expected value of $\rho(v)$ is minimized:

$$E(\rho(v)) = \int_{-\infty}^{+\infty} \rho(v)p(v)dv \rightarrow \min, \tag{1}$$

where $\rho(\cdot)$ is a "loss" function, $E(\cdot)$ is the risk (the expected value of the loss), and $p(\cdot)$ is the probability density function of v. This means that the transformed representation $\mathbf{v}$ should be less redundant (because of $n \leq m$) and salient (because of $E(\rho(v)) \rightarrow \min, v \in \mathbf{v}$). Thus $E(\cdot)$ is a measure of saliency which depends on the loss function $\rho(\cdot)$.

A simple example of the representation transformation is the linear mapping $\mathbf{W}$ which transforms the m-dimensional input representation $\mathbf{x}$ to n-dimensional output representation $\mathbf{v}$ by using

$$\mathbf{v} = \mathbf{W}\mathbf{x} = (\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_m)^{\mathsf{T}}\mathbf{x}. \tag{2}$$

In this case, $\mathbf{W}$ is a feature extractor if $\mathbf{v}$ has some nice properties. The feature extractor $\mathbf{W}$ can be realized by using a single-layer linear feed-forward network and its basic unit is a m to 1 mapping

$$v = \mathbf{w}^{\mathsf{T}}\mathbf{x} = \mathbf{x}^{\mathsf{T}}\mathbf{w}, \quad v \in \mathbf{v}. \tag{3}$$

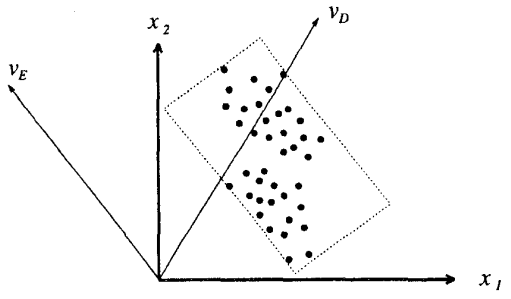

Figure 4. A point cloud in a 2-dimensional space.

The basic unit can also be nonlinear. In this case the m to 1 mapping is formulated by

$$v = \tau(\mathbf{w}^\top \mathbf{x}) = \tau(\mathbf{x}^\top \mathbf{w}), \quad v \in \mathbf{v}, \tag{4}$$

where $\tau(\cdot)$ is nonlinear function. The mapping (3) or (4) is salient or interesting if $E(\rho(v))$ is minimized. The key issue of constructing a feature extractor is thus the design of the loss function. Before the loss function can be designed, the question of which v is "salient" or "interesting" should be first defined. Two general definitions about the saliency of v that we have are:

- **Expressiveness**: v is salient if it is *expressive.*
- **Discrimination**: v is salient if it is *discriminating.*

3.2. Expressive Feature

Let us consider a set of m-dimensional vector $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_t\}$ which builds a "cloud" of points in the m-dimensional space. It is clear that each point $\mathbf{x} \in \mathcal{X}$ can be projected onto a direction determined by the vector $\mathbf{w}$ by using Equation (3) or (4) and the result of this projection is v. Figure 4 just shows a case of $m = 2$. Now the problem is which projection direction is interesting.

As shown in Figure 4, the first interesting direction is v_E because the projection of all points onto this direction has the maximal variance and v_E is, therefore, *expressive.* It can be proved that v_E is determined by that $\mathbf{w}$ which is the largest eigenvector associated with the largest eigenvalue of the correlation matrix $\mathbf{Q} = E(\mathbf{x}\mathbf{x}^\top)$. Let us first define a loss function

$$\rho_G = \frac{1}{2} v^2 \tag{5}$$

The risk $E(\rho_G)$ can be calculated by

$$E(\rho_G) = \frac{1}{2} E(v^2) = \frac{1}{2} \mathbf{w}^\top \mathbf{Q} \mathbf{w}. \tag{6}$$

Minimizing $E(\rho_G)$ requires

$$\Delta \mathbf{w} = \frac{\partial E}{\partial \mathbf{w}} = \mathbf{Q}\mathbf{w} = \mathbf{0}. \tag{7}$$

This leads to famous *plain Hebbian learning* rule

$$\Delta w_i = \eta v x_i, \tag{8}$$

where w_i and x_i are the i^{th} component of $\mathbf{w}$ and $\mathbf{x}$ respectively, Δw_i is the change in w_i, and η controls the learning rate as usual. It can be seen that Hebbian learning is controlled by both the input (through x_i) and the output (through v). It is well-known that there are only unstable fixed points for plain Hebbian learning procedure (8)([6]).

Let us modify the loss function (5) to

$$\rho_E = \frac{1}{2} \left[v^2 - E(v^2) \mathbf{w}^\top \mathbf{w} \right]. \tag{9}$$

The risk $E(\rho_E)$ can be calculated by

$$\begin{aligned} E(\rho_E) &= \frac{1}{2}(E(v^2) - E(v^2)\mathbf{w}^\top \mathbf{w}) \\ &= \frac{1}{2}(\mathbf{w}^\top \mathbf{Q} \mathbf{w} - E(v^2)\mathbf{w}^\top \mathbf{w}). \end{aligned} \tag{10}$$

Since the risk is continuously differentiable, the optimization of (10) can be achieved, via a gradient descent method, with respect to $\mathbf{w}$:

$$\Delta \mathbf{w} = \frac{\partial E}{\partial \mathbf{w}} = \mathbf{Q}\mathbf{w} - E(v^2)\mathbf{w} = \mathbf{0}. \tag{11}$$

Clearly, an equilibrium can be reached if $\mathbf{w}$ is the eigenvector associated with one eigenvalue, say the largest one, of $\mathbf{Q}$ and $E(v^2)$ is just the eigenvalue.

Equation (11) leads to the learning rule suggested by Oja ([6]). According to this rule, each input $\mathbf{x} \in \mathcal{X}$ is applied to adapt the weight $\mathbf{w}$ by using

$$\Delta w_i = \eta v (x_i - v w_i). \tag{12}$$

The learning rule (12) is a generalized version of Hebbian learning rule (8) which has been widely applied to develop unsupervised learning networks.

Comparing the loss function (9) with (5) shows that these Hebbian-like learning rules are based on second order statistics as they usually use second order polynomials for measuring the interest and they lead to extraction of *principal components* (PC) of the input data [5, 6]. It can be proved [5] that minimizing the risk (10) is equivalent to maximizing the information content of the output representation in situations where the data has a Gaussian distribution.

The direction v_E found in this way allows faithful representation of the input data and the projection of the point cloud onto this direction can also show interesting structure if the cloud contains a few clusters and the separation between clusters is larger than the internal scatter of the clusters. However, the direction v_E can lead us astray if the cloud shows too many isotropically distributed clusters or if the data has a high noise level. In these two cases, the output representation v_E doesn't allow discrimination between clusters (see the example in Figure 4). This means that second order polynomials are not sufficient to characterize the important features of an input distribution and all second order statistics based feature extractors cannot provide features which are discriminating enough for recognizing the structure in the input representation.

3.3. Discriminating Feature

As shown in Figure 4, the second interesting direction is v_D because the projections of all points onto this direction can enable us to better distinguish the interesting structure (clusters) presented in the cloud and v_D is, therefore, *discriminating*. In order to find this direction, a measure sensitive to distributions which are far from Gaussian is needed. As already discussed, second order polynomials such as shown in (9) cannot be used for measuring deviation from normality. To emphasize bi- or multi-modality of the projected distribution, higher order polynomials are required and care should be taken to avoid their over-sensitivity to small number of outliers. Let us define a loss function

$$\rho_D = v^2 \left[\frac{E(v^2)}{4} - \frac{|v|}{3} \right] = v^2 r(v), \tag{13}$$

where $r(v)$ can be regarded as a weighting function. It is clear that the loss function ρ_D is small if v is close to zero or to $3E(v^2)/4$. Moreover, it remains negative for $v > 3E(v^2)/4$. Thus, ρ_D as an index can exhibit the fact that bimodal distribution is already interesting, and any additional mode should make the distribution even more interesting.

Actually, any radial basis function (see [8]) can be used as the weighting function in Equation (13) to design interesting loss functions. The advantage of $r(v)$ used in Equation (13) is its connection to the Bienenstock, Cooper, and Munro (BCM) theory of visual cortical plasticity [4].

The expected value of ρ_D is given by:

$$E(\rho_D) = \frac{1}{4}E^2(v^2) - \frac{1}{3}E(v^3) = \frac{1}{4}E(v^2)\mathbf{w}^\top \mathbf{Q}\mathbf{w} - \frac{1}{3}E(v^3). \tag{14}$$

To achieve $E(\rho_D) \rightarrow \min$, the equation

$$\Delta \mathbf{w} = \frac{\partial E}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}} \left[\frac{1}{4}E^2(v^2) - \frac{1}{3}E(v^3) \right] = \mathbf{0} \tag{15}$$

should be satisfied. This leads to a learning rule

$$\Delta w_i = \eta \left[v^2 - E(v^2)v \right] x_i. \tag{16}$$

The difference between the learning rule (16) and Hebbian learning rule (8) is the fact that the influence of the output on the learning process (the feedback) has been changed from v in (8) to $v^2 - E(v^2)v$ in (16). This enables the learning rule (16) to discover bimodal distributions as Δw_i in (16) (unlike in (8)) has opposite value depending on if v is larger or smaller than $E(v^2)$.

Unfortunately, the learning rule (16) has the same divergence problem like Hebbian learning rule (8) and in any case $\mathbf{w}$ does not settle down. One way to prevent the divergence of $\mathbf{w}$ is to constrain the growth of $\mathbf{w}$ by modifying the loss function (13):

$$\rho_Z = v^2 \left[\frac{E(v^2)}{4} - \frac{|v|}{3} \right] - E(v^2)\mathbf{w}^\top \mathbf{w}. \tag{17}$$

This leads to a new learning rule obtained by adding a weight decay to the learning rule (16):

$$\Delta w_i = \eta \left[v^2 - E(v^2)v \right] (x_i - v w_i). \tag{18}$$

So far four different learning rules have been introduced. They are based on four different loss functions and can be applied to extracting expressive and discriminating features.

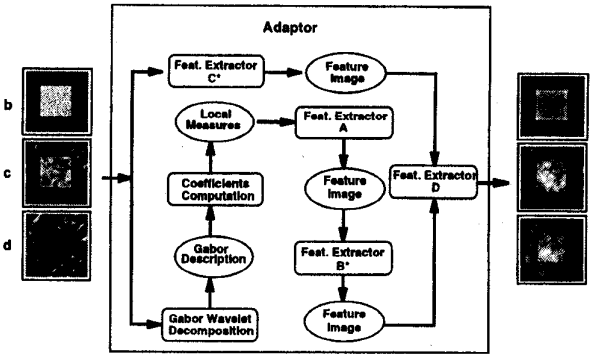

Figure 5. Input adapting for the image thresholding algorithm.

4. Adaptor Design

Figure 5 shows an adaptor which is designed for the image thresholding algorithm. The key idea for designing this adaptor is to decompose the input image into some local measure images and then to adaptively extract salient features from these local measure images based on the modified Hebbian learning rules presented above. In order to derive local measures for each pixel in the input image, the quadrature Gabor filter kernels

$$G_+(\omega,\phi) = \exp\left[-\frac{\lambda^2\omega^2(x^2+y^2)}{4\pi} \right] \cdot \cos[\omega(x\cos\phi + y\sin\phi)] \tag{19}$$

$$G_-(\omega,\phi) = \exp\left[-\frac{\lambda^2\omega^2(x^2+y^2)}{4\pi} \right] \cdot \sin[\omega(x\cos\phi + y\sin\phi)] \tag{20}$$

are applied to decompose the input image $I(x,y)$ by using

$$I_+(x,y,\omega,\phi) = G_+(\omega,\phi) * I(x,y), \tag{21}$$

$$I_-(x,y,\omega,\phi) = G_-(\omega,\phi) * I(x,y), \tag{22}$$

where ω and ϕ are the modulation (center) frequency and orientation, respectively, of the Gabor filter kernel; λ is the ratio of the channel bandwidth and the modulation frequency; and $I_+(x,y,\omega,\phi)$ and $I_-(x,y,\omega,\phi)$ are Gabor space image descriptions. From these descriptions it is easy to derive some local measures. In Figure 5 the power

$$p(x,y,\omega,\phi) = I_+^2(x,y,\omega,\phi) + I_-^2(x,y,\omega,\phi) \tag{23}$$

is used as a local measure of the input image $I(x,y)$. So far m power images can be obtained and m depends on the quantization of ω and ϕ. This means a local measure vector with m elements is associated with each pixel of the input image.

Each element of the local measure vector is a representation to describe a local property of the input image but it is not just the right feature to discriminate clusters depicted in the input image. The most discriminating feature should be found in the m-dimensional

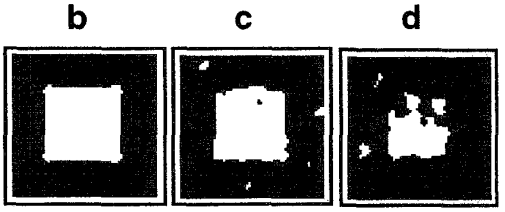

Figure 6. Improving the performance of the image thresholding algorithm.

Figure 7. Sample test result using SAR images.

local measure space based on the structure presented by all local measure vectors in the input image. This requires a m to 1 feature extractor **A** which is trained by using the learning rule (16) or (18) as described above.

To reduce high frequency components in the input data two feature extractors **B**$*$ and **C**$*$ are introduced into the adaptor shown in Figure 5. They are actually two convolution kernels with $n \times n$ elements which should be trained by using the learning rule (8) or (12) as described above. After the convolution using **B**$*$ and **C**$*$ two feature images can be produced which should be integrated by the feature extractor **D** in order to supply desired images for the thresholding algorithm. The feature extractor **D** performs a 2 to 1 transformation and is trained by using the learning rule (16) or (18).

5. Experimental Results

The adaptor shown in Figure 5 is able to produce desired input images for the thresholding algorithm. The output images in Figure 5 are better than the input images in Figure 5 because the object in the center is better discriminated from the background. Thus, the performance of the thresholding algorithm can be improved by using the adaptor. Figure 6 shows the performance of the thresholding algorithm with the adaptor. Obviously, the object in the center is better discriminated from the background even if the input image **d**, which is not desired by the thresholding algorithm (see Figure 1), is presented.

Figure 7 shows the test result of object detection system using real SAR image data. The image **a** shows the input image. The image **b** shows the test result using the thresholding algorithm.

Figure 8. Sample test result using a FLIR image.

The image **c** shows the test result using the thresholding algorithm plus the input adaptor. Figure 8 shows another example of target detection in a FLIR (Forward Looking Infrared) image by using the same system. Again, the image **a** is the input image. The image **b** and **c** show the test results using the thresholding algorithm without and with the input adaptor.

6. Conclusions

In this paper, the attention was paid on how to improve the performance of an object detection system by adding the adaptability to ready-made available algorithms without changing their internal structure. Other researchers can make use of the approach and the results presented in this paper for 1) information fusion and integration and 2) robust object detection/recognition.

Acknowledgment: The help of Yeli Chen in preparing this paper is gratefully acknowledged.

References

[1] B. Bhanu and S. Lee. *Genetic Learning for Adaptive Image Segmentation*. Kluwer Academic Publishers, 1994.

[2] Y. Chen. Vergleichende untersuchung neuronaler netze zur dimensionsreduzierung. Master's thesis, Fakultäte Informatik, University of Stuttgart, Germany, 1995.

[3] R. M. Haralick. Performance characterization protocol in computer vision. In *Proc. Performance Versus Methodology in Computer Vision*, pages 26–32, Seattle, WA, June 1994.

[4] N. Intrator and L. N. Cooper. Objective function formulation of the BCM theory of visual cortical plasticity: Statistical connections, stability conditions. *Neural Networks*, 5:3–17, 1992.

[5] R. Linsker. Self-organisation in a perceptual network. *Computer*, 21(3):105–117, 1988.

[6] E. Oja. A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15:267–273, 1982.

[7] Y.-J. Zheng. Feature extraction and image segmentation using self-organization networks. *Machine Vision and Applications*, 8(5):262–274, 1995.

[8] Y.-J. Zheng, W. Ritter, and R. Janssen. An adaptive system for traffic sign recognition. In *Proc. Intelligent Vehicles Symposium*, pages 165–170, Oct. 1994.